An OpenCL ray tracing approach for Simulating Electromagnetic Fields

Ir. Filip Nauwelaerts

Lab & Quality Manager - Laboratoria De Nayer

PHPC2012 - 13 december 2012

Content

- Context description
- An adapted sequential ray tracing algorithm
- First attempt to accelerate by implementing parallel computing

Context description

Real life situations

Verification by measurement

- High electric field strengths E [V/m]
- Unpredictable environment
- Risk of failure or loss of performance
- Predictable, controlled and repetitive
- According to standard:
  - MIL-STD-461E RS103
  - RTCA DO160E
  - EN61000-4-21
- SAR, FAR, OATS, **Reverberation chamber**

RVC Loading and field uniformity verification

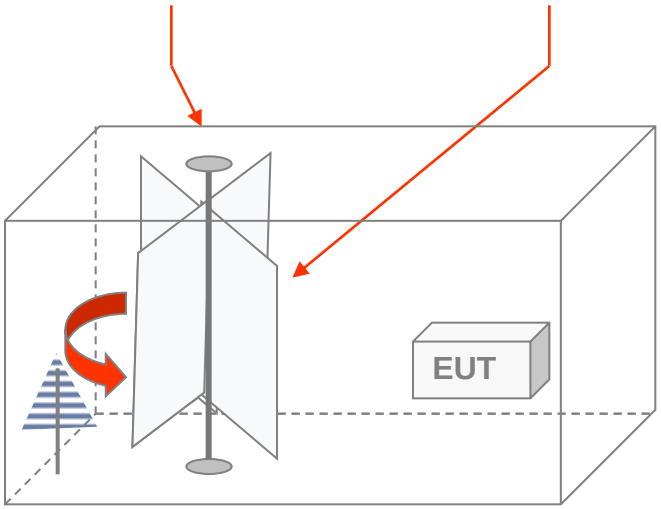
- Goal: High field strength with moderate input power,

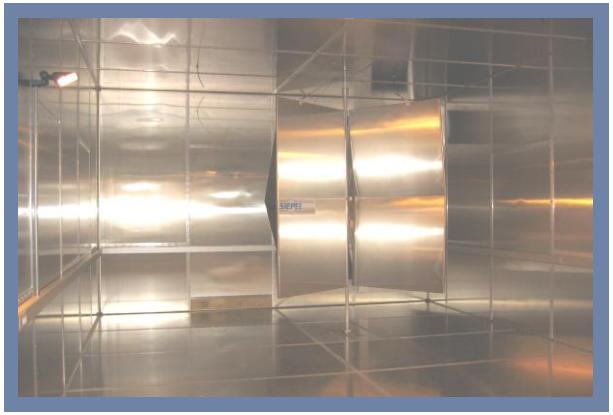
statistically homogeneous and with random polarization

High reflection rate
=> High Q factor,
=> standing waves

Continuously changing
boundary conditions

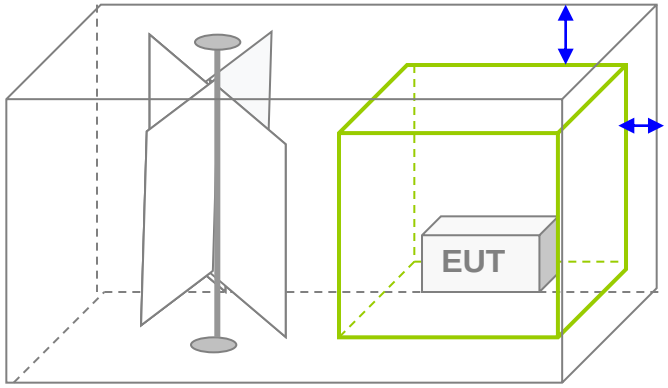
- RVC Characteristics:
  Overmoded highly conductive cavity with mode stirrer





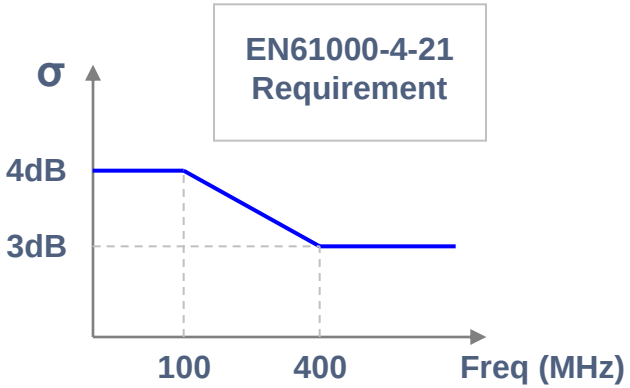
RVC Loading and field uniformity verification

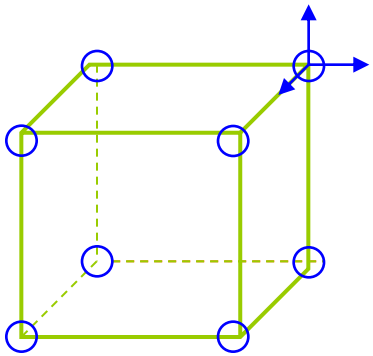
• **Virtual test volume:** minimum distance between wall and EUT ~ λ_{LF} *(EN61000-4-21: 1/3 . λ_{LF})*



• **Empty chamber field uniformity:**

8 corner points,
3 field values $|E_x|$, $|E_y|$ and $|E_z|$ (max over 2π stirrer rotation)
=> 24 values in total





RVC Loading and field uniformity verification

- Influence of EUT on field uniformity:

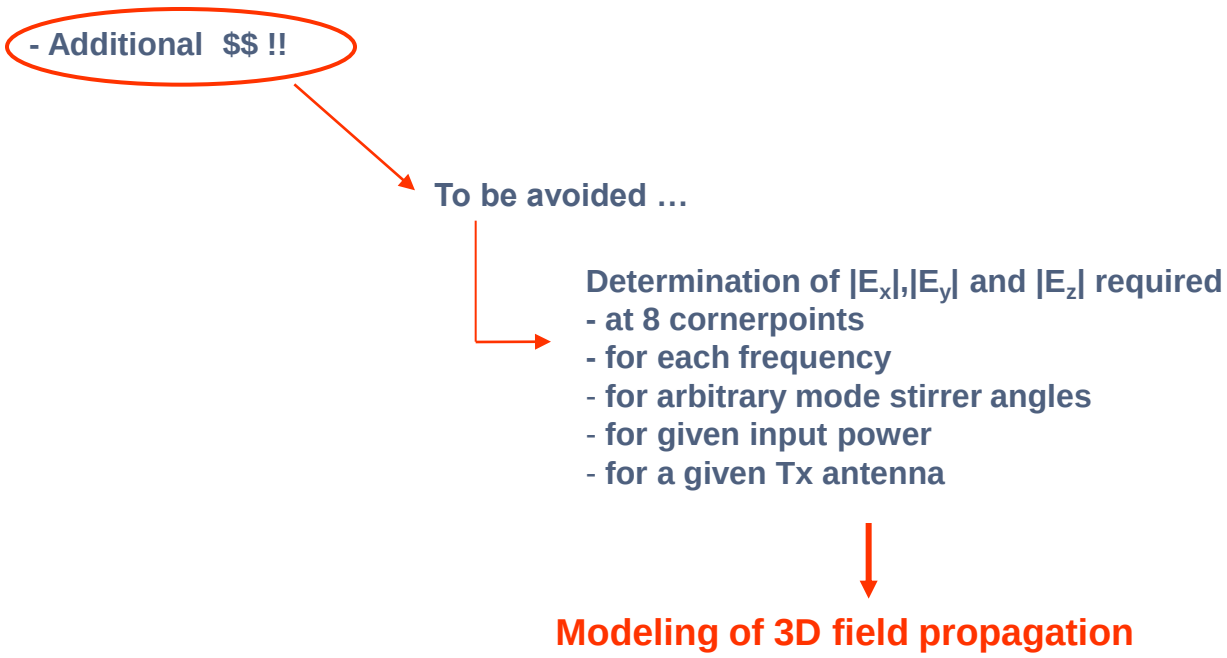
  - EUT absorption/reflection => chamber loading may exceed maximum

  - Field uniformity requirements (low statistical field deviation) exceeded ?

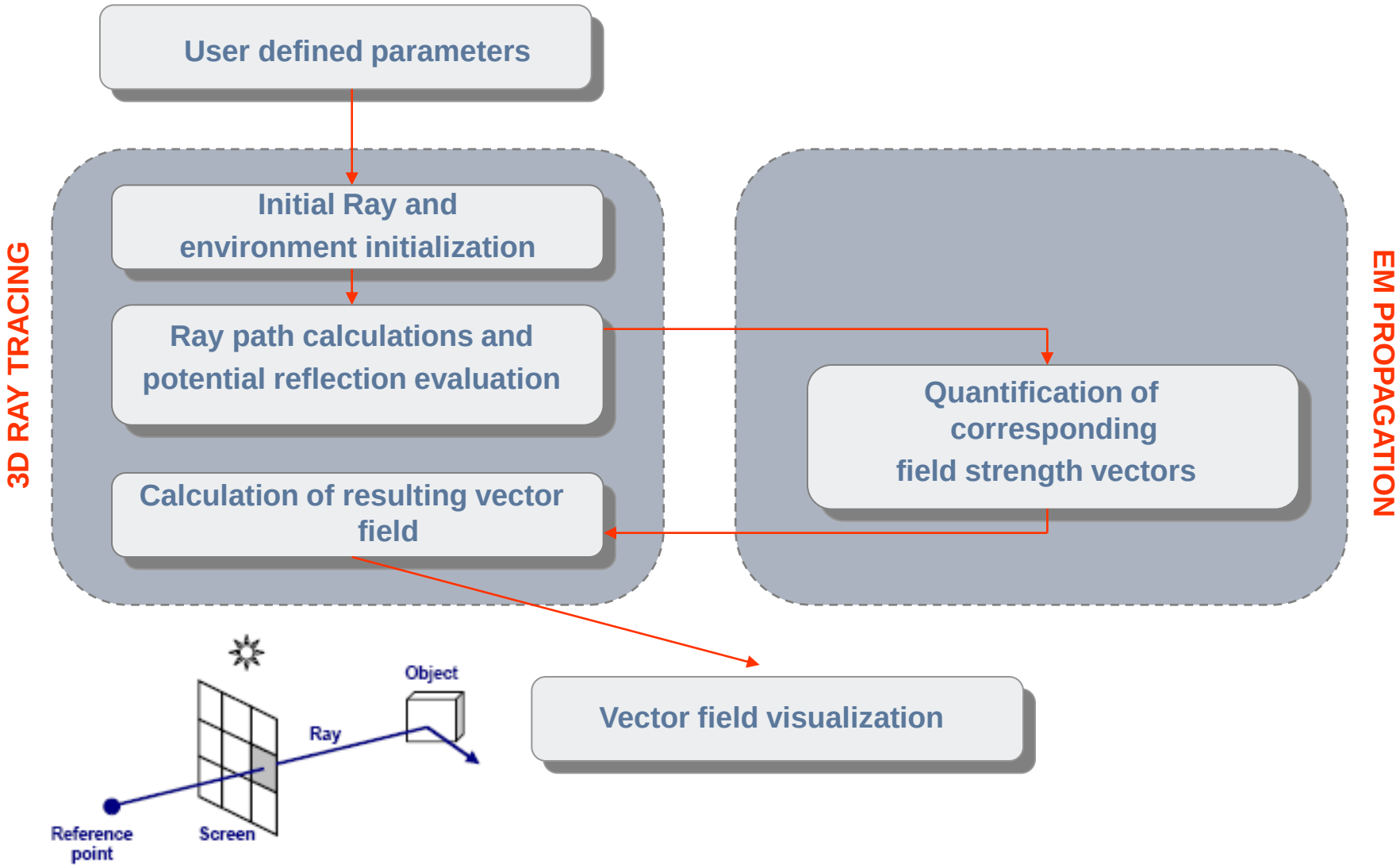
- Preliminary chamber loading verification needed (with EUT in place) (CLF: EN61000-4-21, Annex B, par.B.2)

  - Additional test time (equipment, test engineer, EUT availability?)

  - Additional $$!!



Adapted Ray Tracing



a) Basic Ray Tracing

- **Ray equation**

$$p = p_0 + s \cdot \vec{u}$$

- **Plane equation**

Given non-collinear points p_0, p_1, p_2 on plane ς in 3D Cartesian coordinate system,

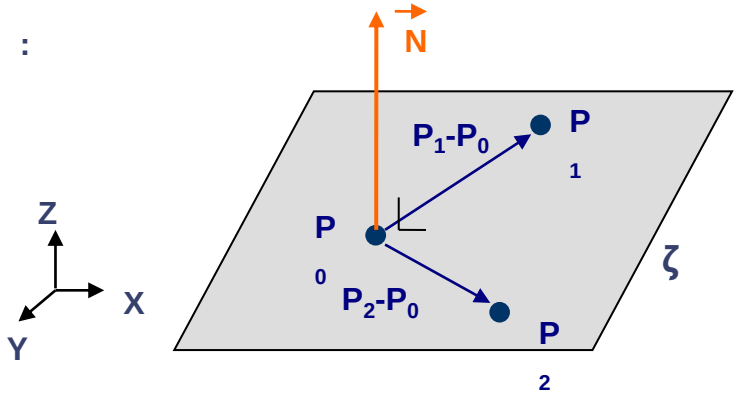
plane equation can be derived ∀ p(x,y,z) ∈ ς :

$$\vec{N} \bullet \vec{P} = -D$$

with $\vec{N}$: surface normal

$\vec{P}$: point on plane ς

D : plane specific C^te

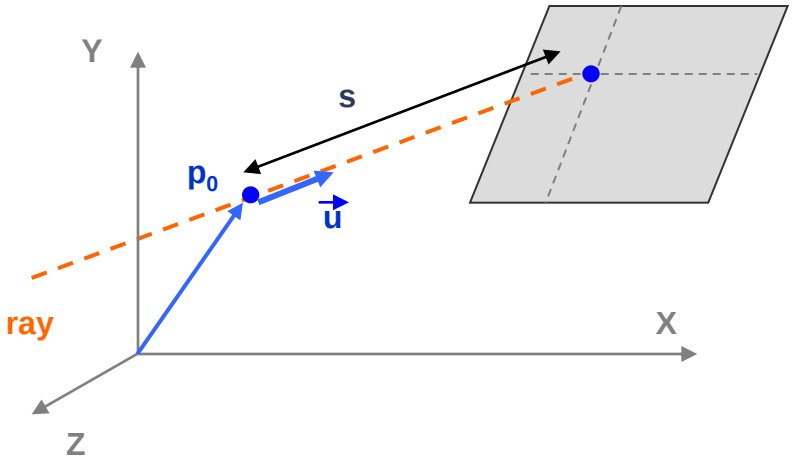


- **Ray – surface intersection**

$$\begin{cases} P = P_0 + s.\vec{u} \\ \vec{N} \bullet P = -D \end{cases}$$

$$\equiv \quad \vec{N} \bullet (P_0 + s.\vec{u}) = -D$$

$$\equiv \quad s = \frac{-D - \vec{N} \bullet P_0}{\vec{N} \bullet \vec{u}}$$



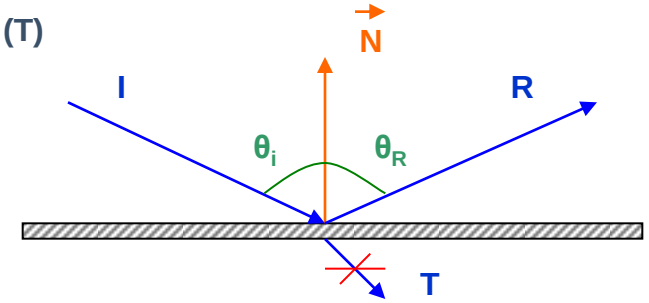
Adapted Ray Tracing

- **Ray reflection**

Reflection of EM waves on panel surface

Specular reflection is assumed (no diffuse spreading): law of regular reflection: $\theta_i = \theta_r$

No refraction (T)



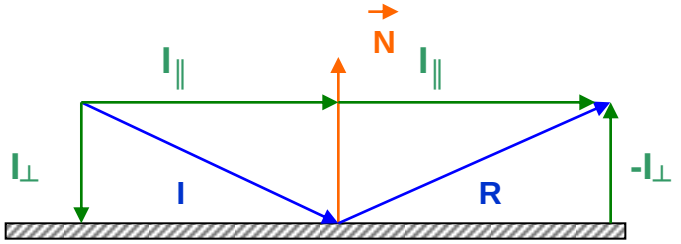
Calculation of reflected ray equation R needed:

$$\vec{R} = (\vec{I} - (\vec{I} \bullet \vec{N}).\vec{N}) - (\vec{I} \bullet \vec{N}).\vec{N}$$

($I_{\parallel}$ and $I_{\perp}$ are the components of $\vec{I}$ parallel and perpendicular to the surface)

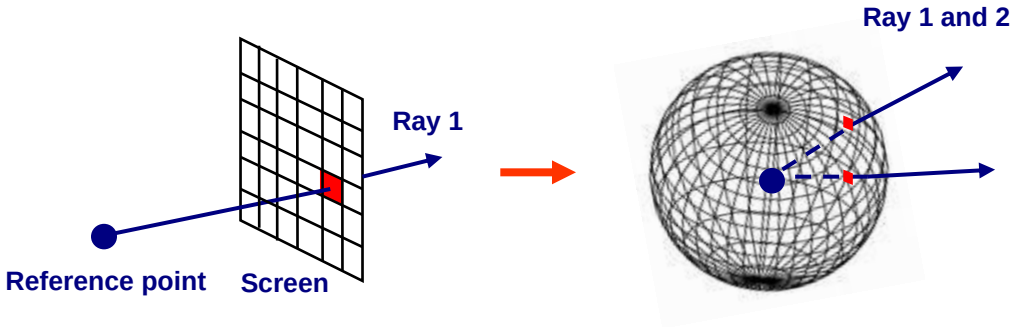
$$\equiv$$

$$\vec{R} = \vec{I} - (2.\vec{I} \bullet \vec{N}).\vec{N}$$



b) Adaptation of standard Ray Tracing

- Multiple viewpoints instead of one

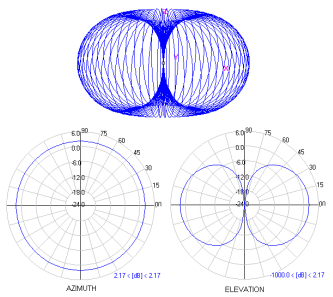


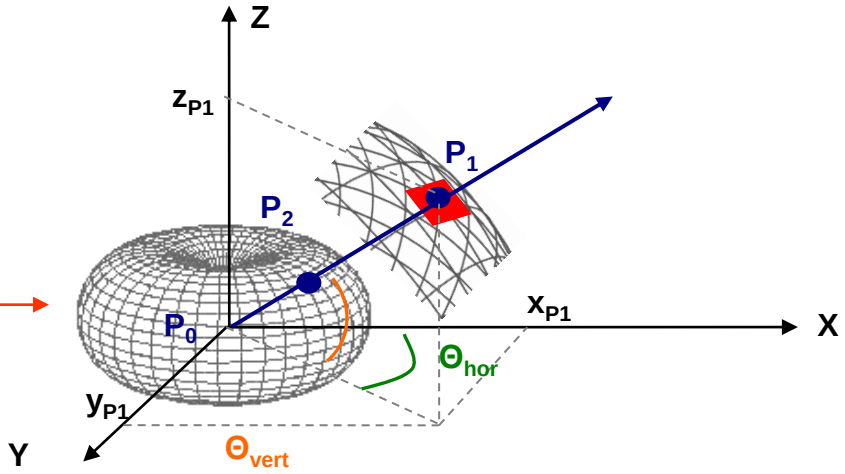
Basic ray tracing	Adapted ray tracing
1 viewpoint => 1 ray / pixel	multiple omnidirectional (or isotropic) viewpoints (inside virtual volume)
Ray origin: viewpoint	Ray origin: Transmit antenna (alternative backtracking algorithm)
No attenuation No phase information	attenuation and phase calculation

Adapted Ray Tracing

• **Adjustment 1**: ray origin = source antenna

Allows 3D interpolation of radiation pattern (dBi)





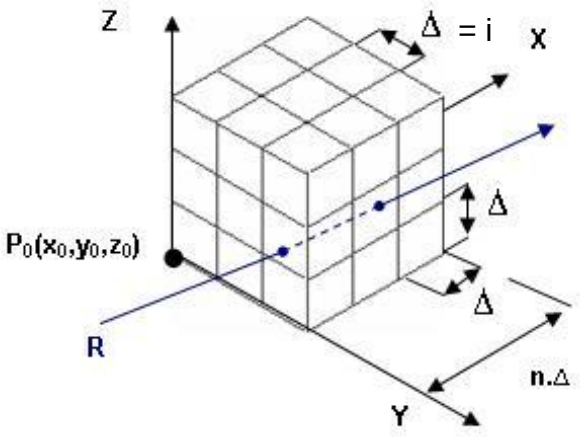
• **Adjustment 2**: Spatial subdivision needed

due to initial angle step at ray origin

=> Association of each viewpoint with cubic volume

avoids missing viewpoints

when minimum angle $\alpha = 2.\arcsin(\frac{i/2}{n.L})$



REMARK: avoid taking same wavefront into account more then once

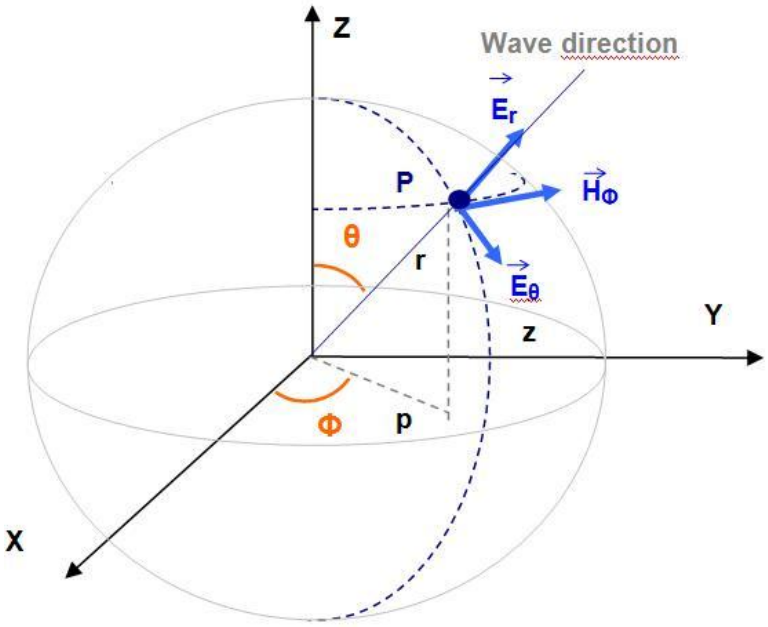
Adapted Ray Tracing

- **Adjustment 3:** **Quantification of EM field strength vectors**

  - **Amplitude and phase:** **related to total ray traveling distance (r)**

  - **For far field, free space, tuned dipole:**

$$\left|\vec{E}_\theta\right| = j.\frac{Z_0.I_m}{2\pi r}.e^{-j.k.r}.\left[\frac{\cos\left(k.\frac{L}{2}.\cos\theta\right)-\cos\left(k.\frac{L}{2}\right)}{\sin\theta}\right]$$



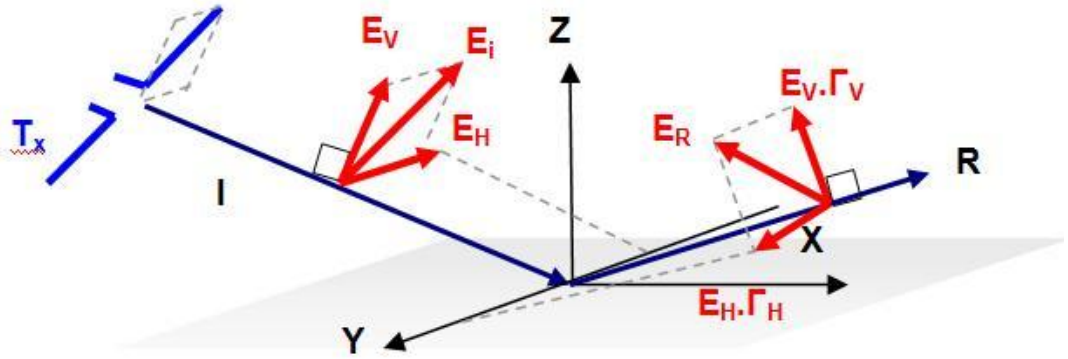
$$\frac{Z_0.I_m}{2\pi r}$$ **amplitude (~ 1 / r)**

$$j.e^{-j.k.r}$$ **phase (~ r)**

$$\left[\frac{\cos\left(k.\frac{L}{2}.\cos\theta\right)-\cos\left(k.\frac{L}{2}\right)}{\sin\theta}\right]$$ **tuned dipole radiation pattern (F(θ))**

Adapted Ray Tracing

• **Adjustment 4:** EM wave reflection



$$\Gamma_H = \frac{\cos\theta - \sqrt{N_d^2 - \sin^2\theta}}{\cos\theta + \sqrt{N_d^2 - \sin^2\theta}}$$

$$\Gamma_V = \frac{N_d^2\cos\theta - \sqrt{N_d^2 - \sin^2\theta}}{N_d^2\cos\theta + \sqrt{N_d^2 - \sin^2\theta}}$$

• **reflection coefficients approximation:** $\Gamma_H = -1$ and $\Gamma_V = 1$ for ideal conducting material

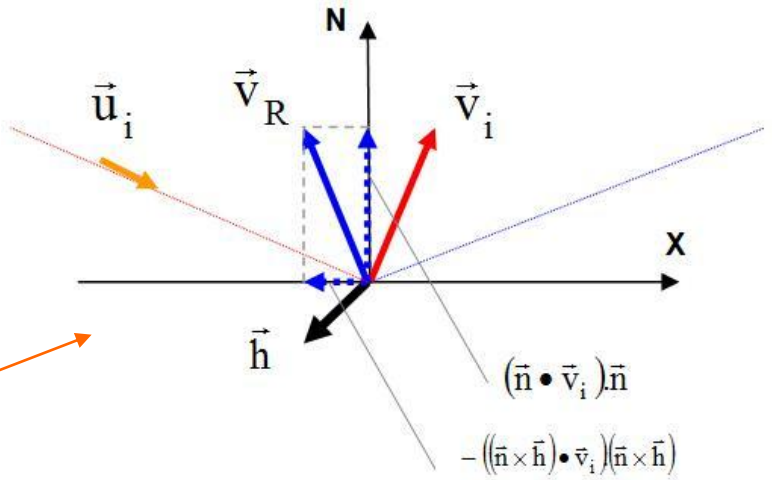
• **Vector reorientation after reflection:**

$$\vec{E}_R = \left|\vec{E}_H\right|.\Gamma_H.\vec{h} + \left|\vec{E}_V\right|.\Gamma_V.\vec{v}_R$$

$$\left|\vec{E}_H\right| = \vec{E}_i \bullet \vec{h}$$

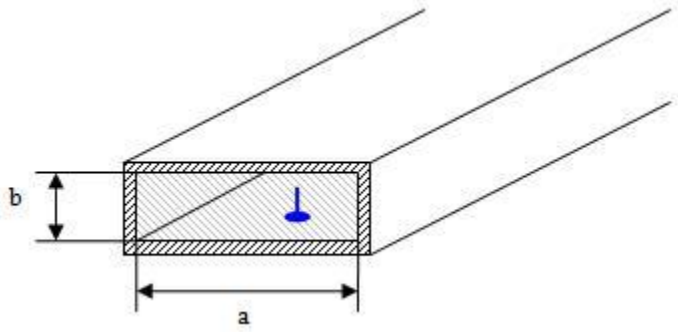
$$\vec{E}_V = \vec{E}_i - \vec{E}_H$$

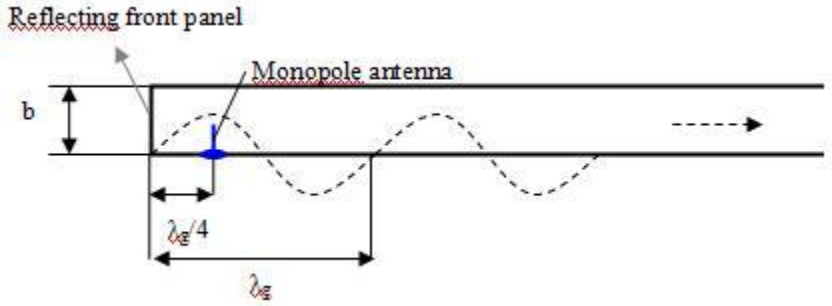
$$\vec{v}_R = (\vec{n}\bullet\vec{v}_i)\vec{n} - ((\vec{n}\times\vec{h})\bullet\vec{v}_i)(\vec{n}\times\vec{h})$$

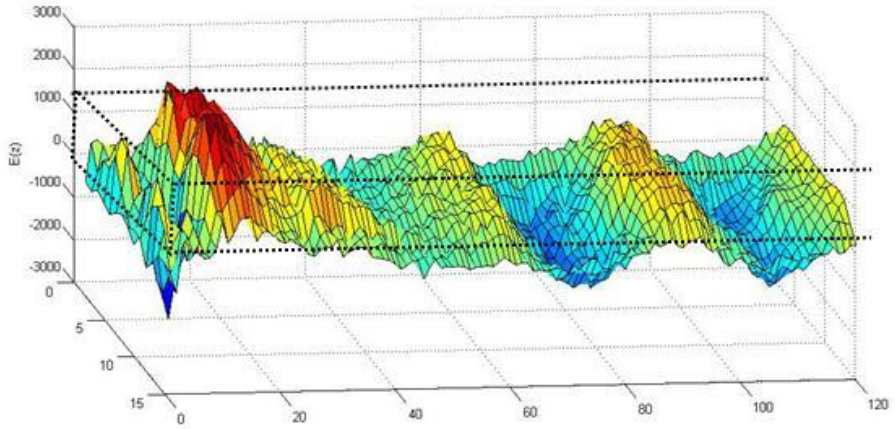


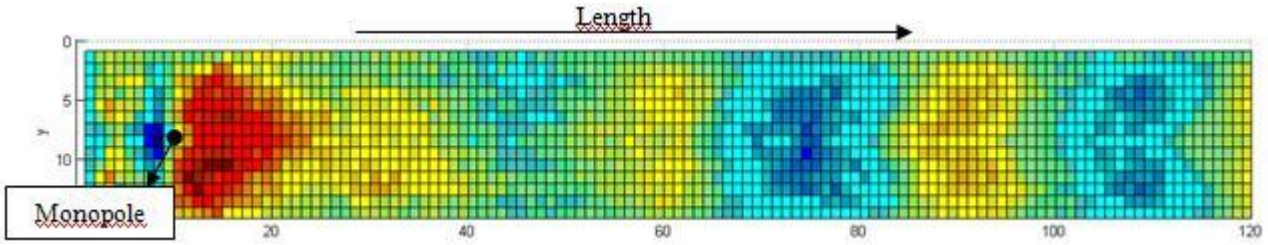
Example after visualization

- TE_{10} waveguide



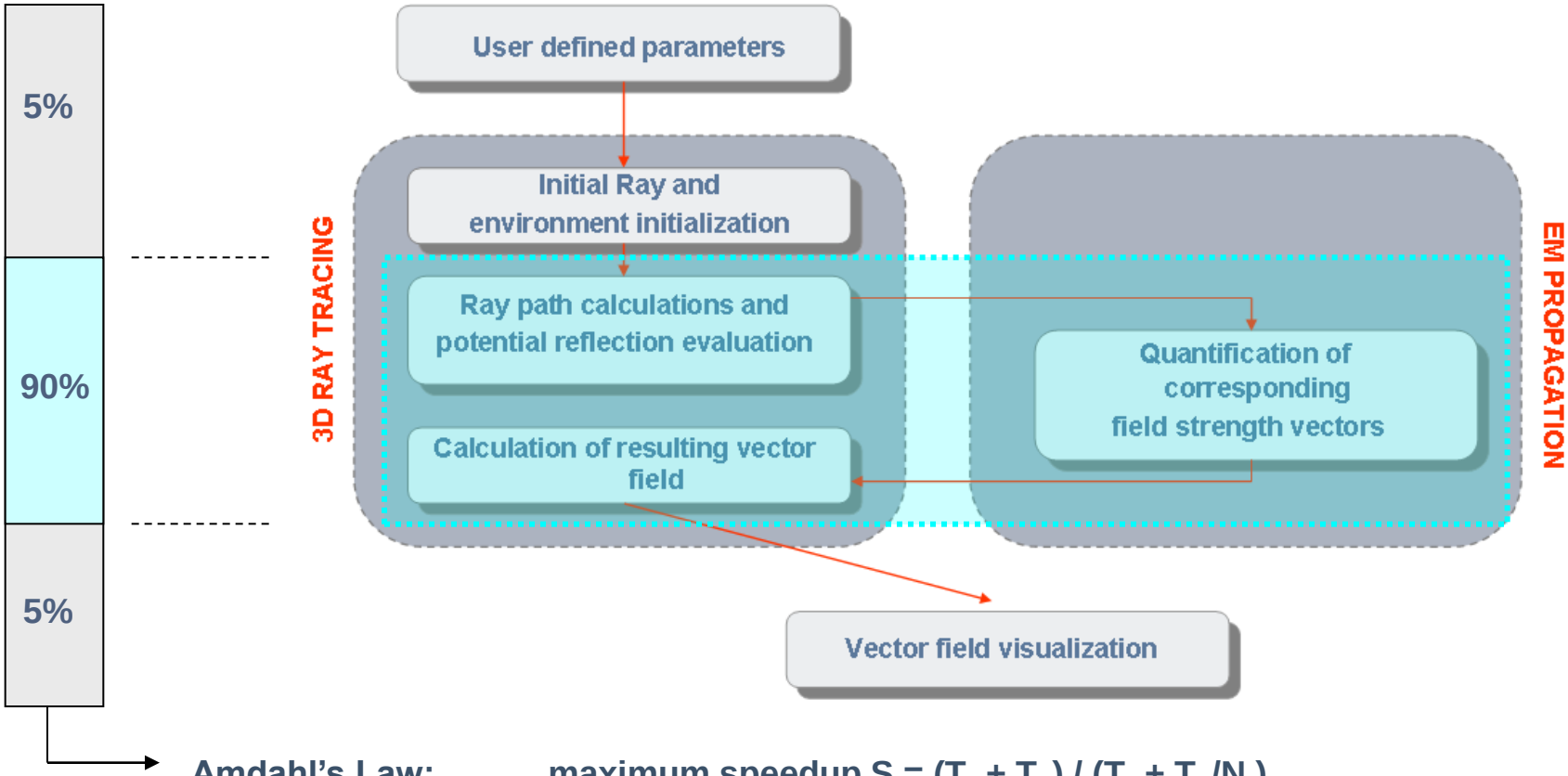






OpenCL implementation

- Can parallel computing be a useful acceleration technique?
  ⇒ which code can be run in parallel?



Amdahl's Law: maximum speedup $S = (T_s + T_P) / (T_s + T_P/N)$
$= 1 / y$ with $y = 10\%$
$= 10$

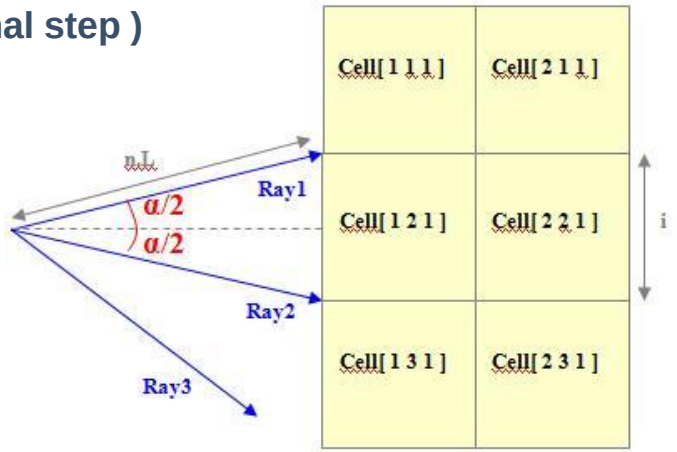
Acceleration techniques

- **Data parallelism** more suitable then Task parallelism
  - Each ray has **individual** contribution to 3D vector field
  - no **data dependency** in between ray calculations
    (Resulting vector field is determined as separate final step)

- **Memory consumption** related to **accuracy**

$$\alpha = 2.\arcsin\left(\frac{i/2}{n.L}\right)$$ where i = cell size ,

n = max number of reflections,
L = longest chamber dimension (3D diagonal between two opposite corners).



"Poor" approach:

a) 1 GHz => i = λ / (2.10) = 0,003m (10 samples per wavelength)
cavity 5m x 5m x 10m => L = 11,45 m
n = 10

α = $75^{E-5°}$ => # initial rays: 173^{E06}
for each ray: 3 floats => 12 bytes x 173^{E06}
= ca.2,07 GB

b) for each viewpoint: 2 floats => e.g. 9m³ = 8 bytes x 10E6 = ca.295 MB

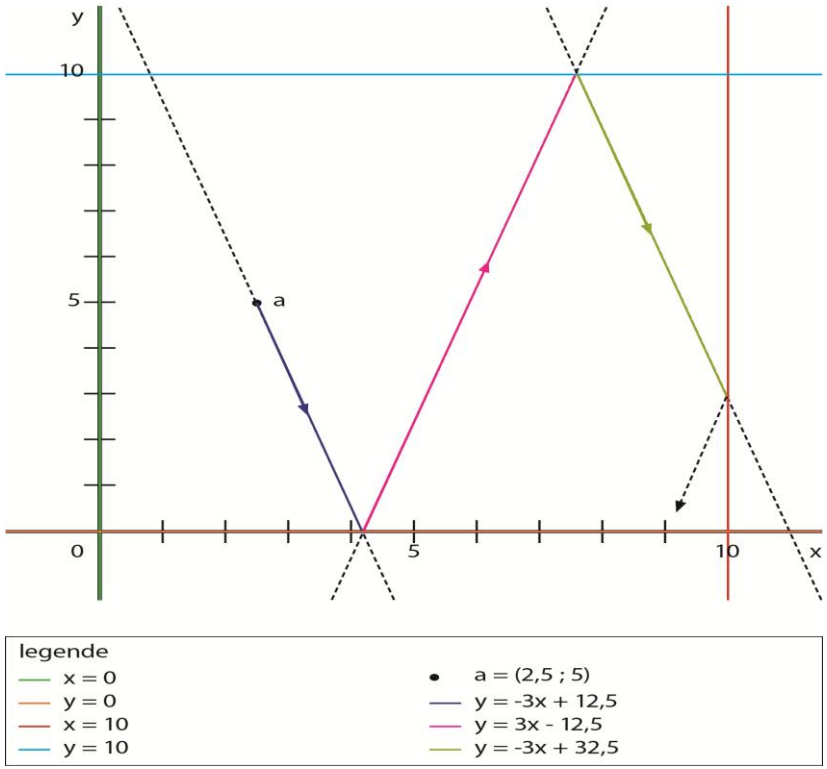
Acceleration techniques

- **Parallel calculation of a limited set of ray tracers**

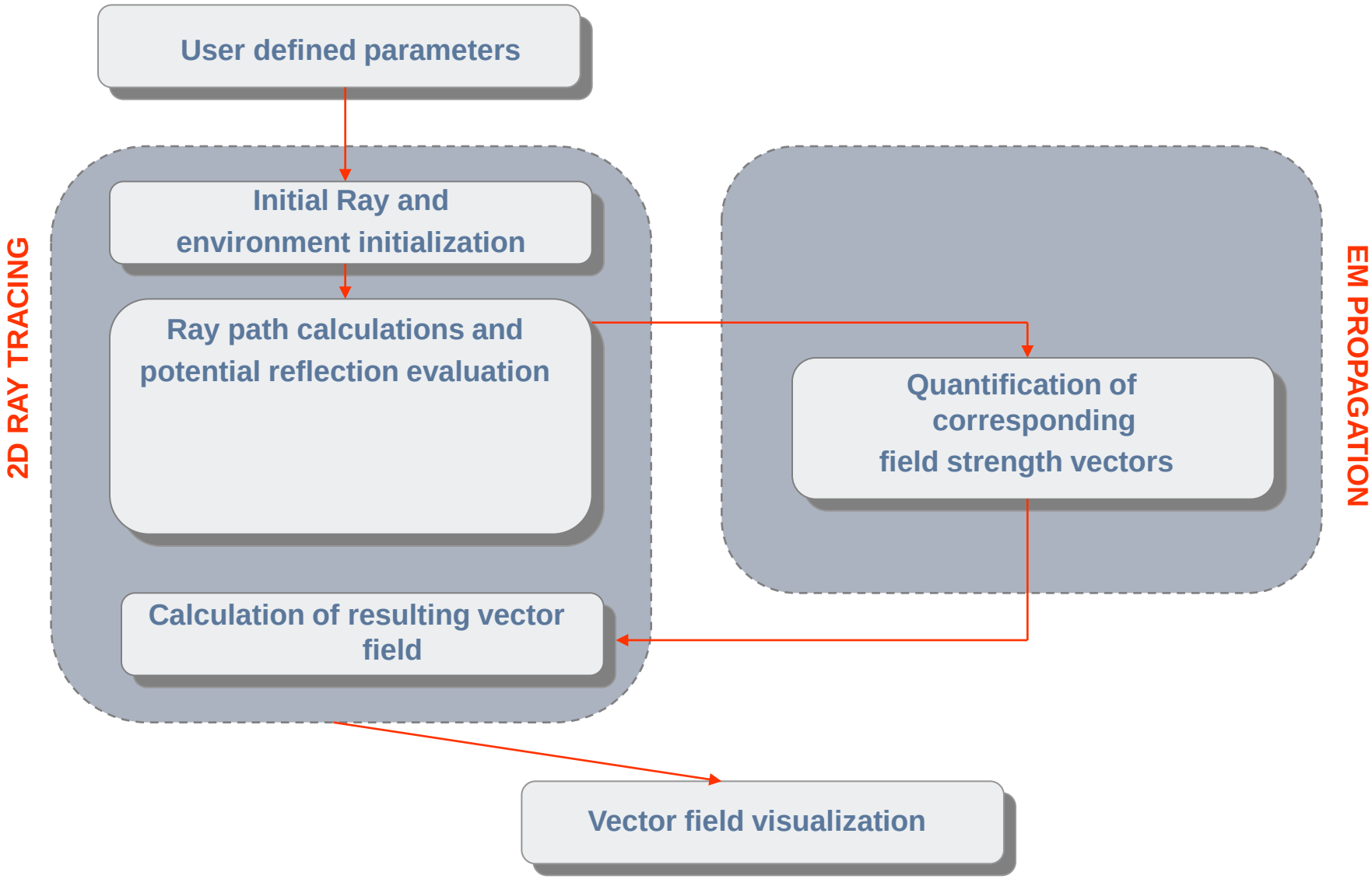
  - **Number of rays per set depending on # processing units**

  - **Calculate contribution of full ray path of each set, then continue with next set**
    => avoids memory overflow for intermediate results

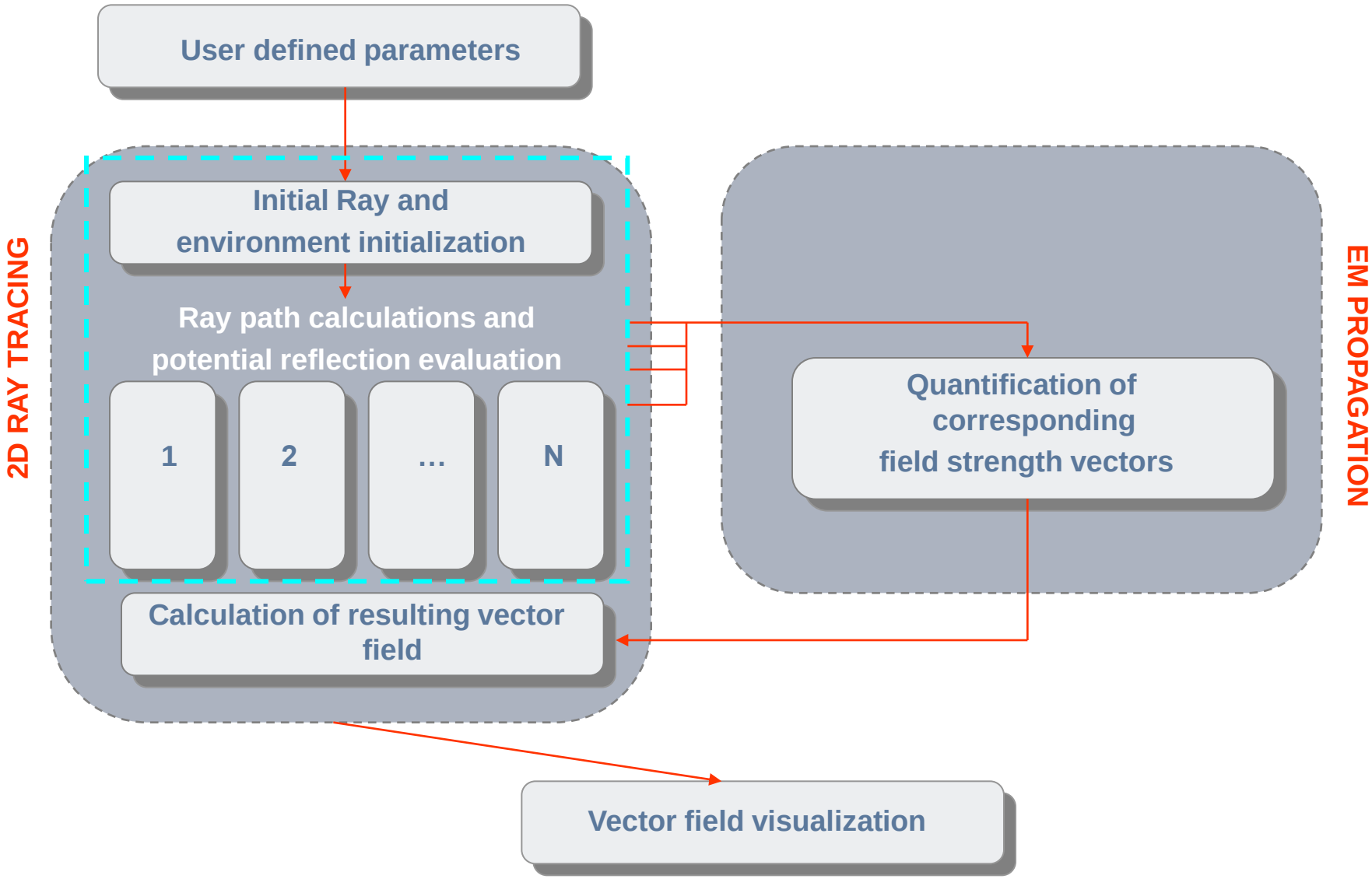
- **2D Implementation** ***(ref: ing. Cedric Busschots)***



Acceleration techniques



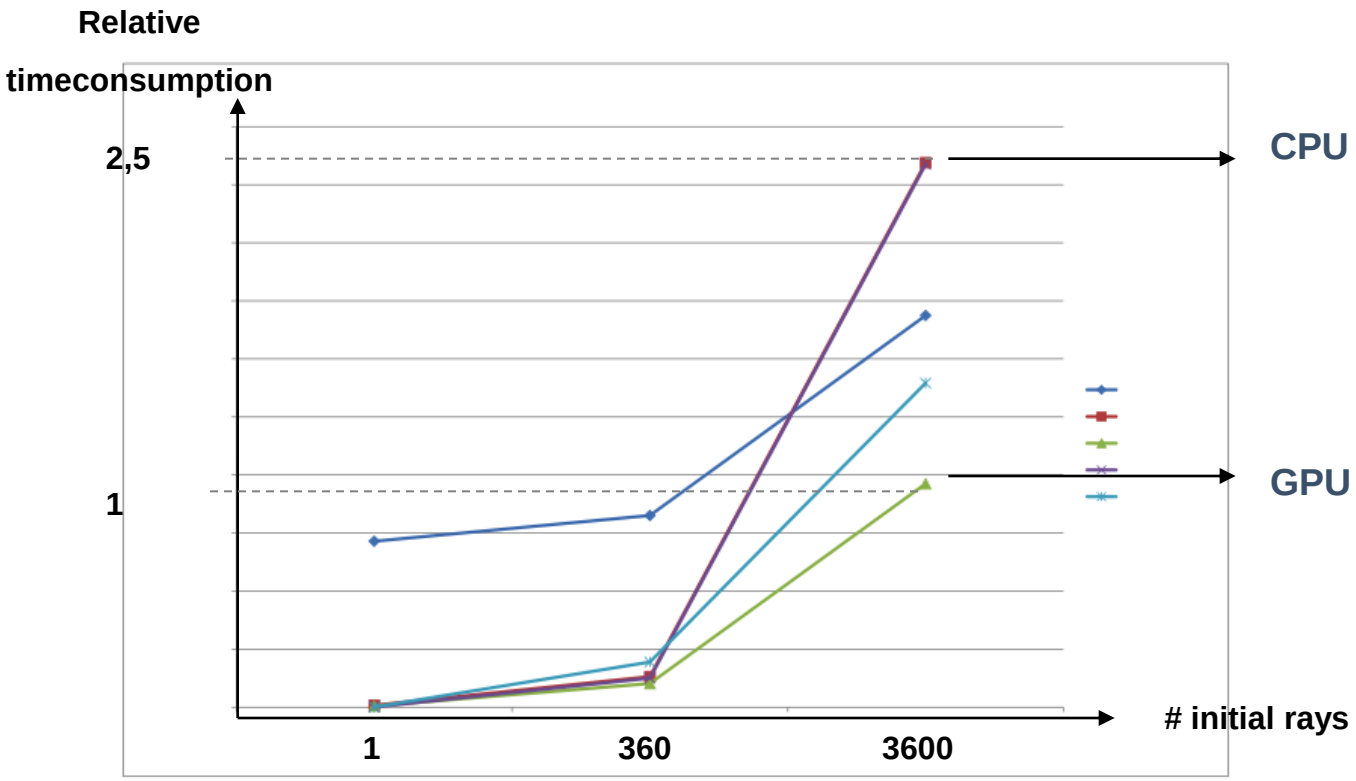
Acceleration techniques



- 2D Implementation – performance analysis *(ref: ing. Cedric Busschots)*

  - Run time comparison for CPU and OpenCL algorithm
  - Timing exercise

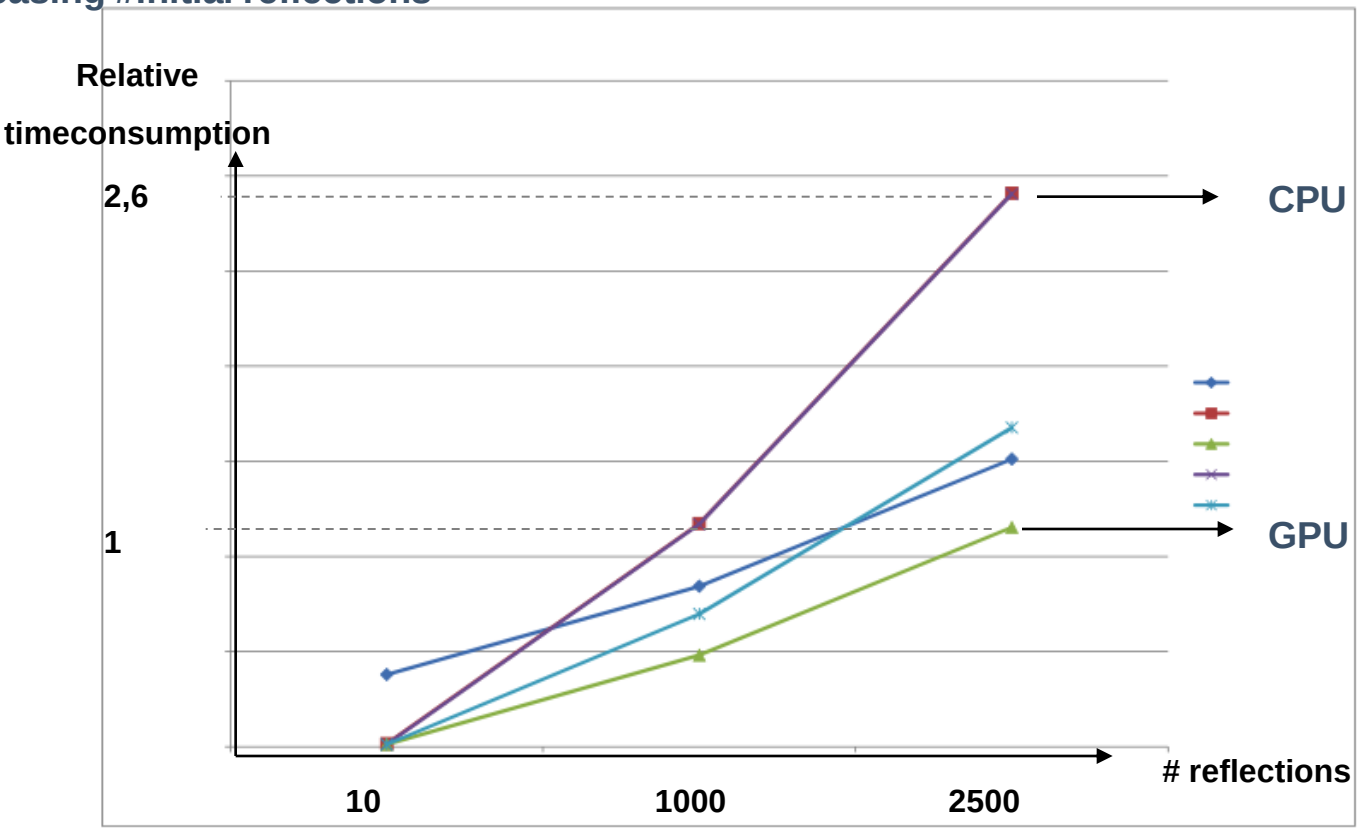
- Increasing #initial rays (1000 reflections)



- 2D Implementation – performance analysis *(ref: ing. Cedric Busschots)*

  - Run time comparison for CPU and OpenCL algorithm
  - Timing exercise

- Increasing #initial reflections



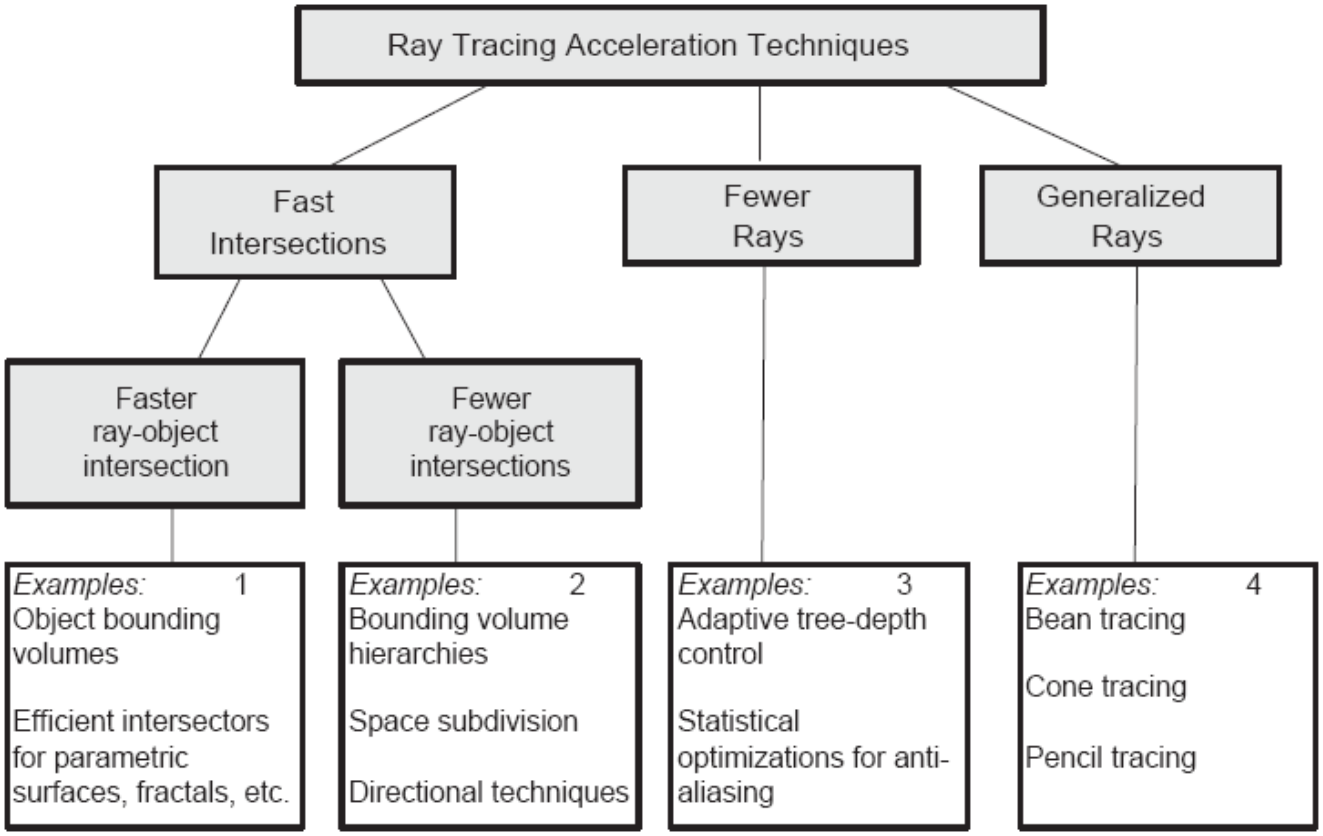
Acceleration techniques

- First conclusions
  - Speedup factor aprx. 2,5
  - Limiting boundaries are memory consumption
  - Further implementations needed to cover:
    - 3D algorithm
    - vector calculus for polarisation and |E| field magnitude
    - algorithm acceleration techniques

Ray tracing acceleration techniques

- **Verschillende technieken beschikbaar,**
  echter vaak gekoppeld aan (dynamische) visualisatie toepassingen

- **Generieke classificatie van acceleratietechnieken:**



Faster and fewer intersections

- ca. 95% van rekentijd bij ray tracing wordt aan intersectie onderzoek besteed
- Complexiteit van standaard ray tracing intersection algoritme:

elke ray wordt met elk object vergeleken

| I | x | O |

I = # initial rays (e.g. 13 x 10^6)
O = # objects (e.g. 10^9)

=> O(n)

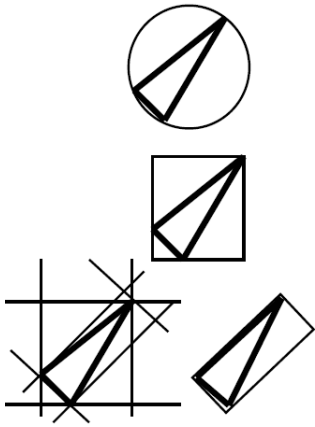
= complexiteit ten gevolge van sequentiële zoektocht naar individuele ray – object intersecties

= lineair zoekalgoritme

= worst case..!!

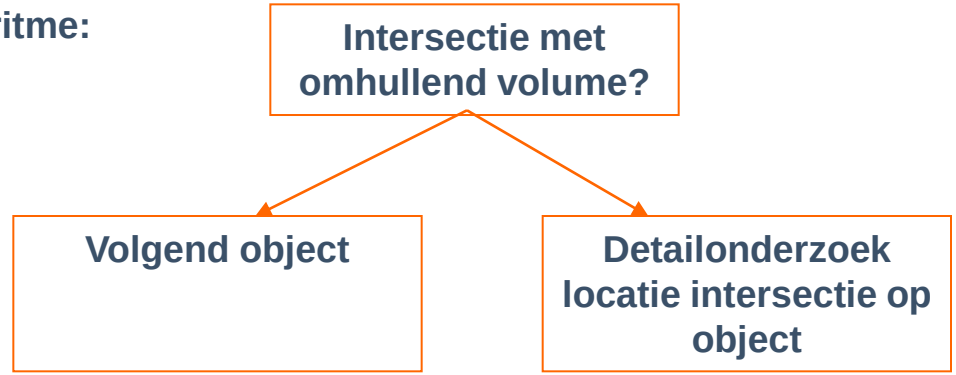
Faster and fewer intersections

- Snellere intersectie-algoritmes:
  - **Object bounding volumes** hanteren:



Objecten worden omhuld door eenvoudige geometrische volumes
vb: kubus, bol, slabs (begrenzende vlakken), …

algoritme:



Snellere berekening, afhankelijk van selectie omhullend volume

vb: Bol volume: $$d = (\mathbf{l} \cdot \mathbf{c}) \pm \sqrt{(\mathbf{l} \cdot \mathbf{c})^2 - \mathbf{c}^2 + r^2}$$

l: ray unit vector, c: center, r: radius

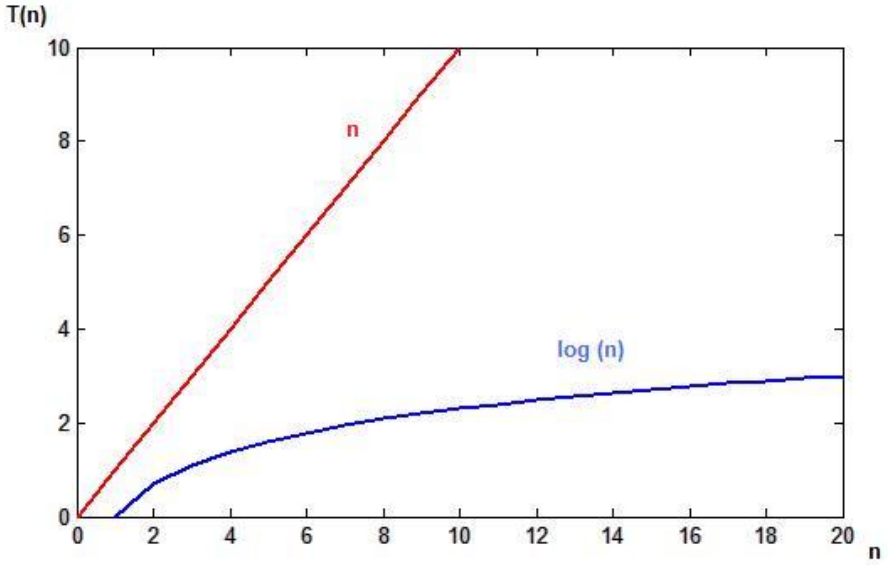
Omhullend volume dient zo nauw mogelijk aan te sluiten aan objectgrenzen

Faster and fewer intersections

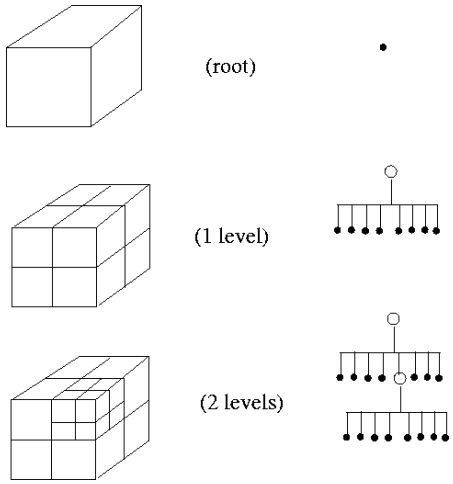
- Reductie aantal te onderzoeken intersecties:
  - Object bounding volumes **hiërarchie** hanteren:

    kleine bounding volumes worden hierarchisch georganiseerd in nieuwe bounding volumes

    Tijdens zoekalgoritme wordt de hiërarchische structuur gevolgd
    => complexiteit O(log n)



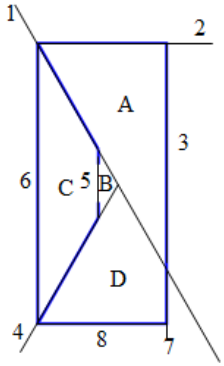
Faster and fewer intersections

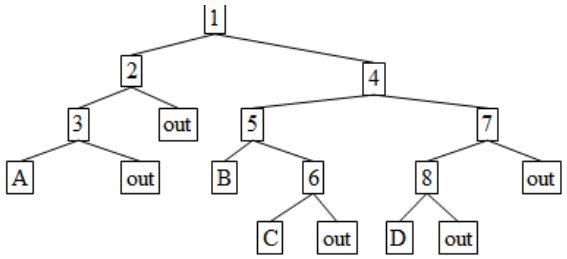


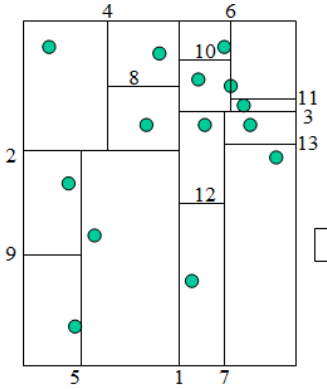
- Reductie aantal te onderzoeken intersecties:
  - Object bounding volumes **hiërarchie** hanteren:

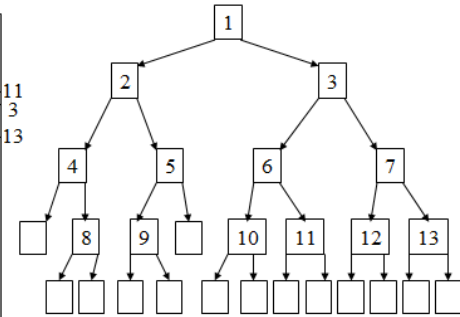
Voorbereiding nodig: recursief opbouwalgoritme

1) initialisatie bij algemeen omhullend volume
2) Opdeling voorgaand volume in deelvolumes
   - Octree
   - BSP (Binary Space Partition):
     voorgaande scene-partitie halveren -> binary search tree
   - kD-tree (k-dimensionale boom) = axis aligned BSP
     objectverdeling in functie van vereiste detaillering object









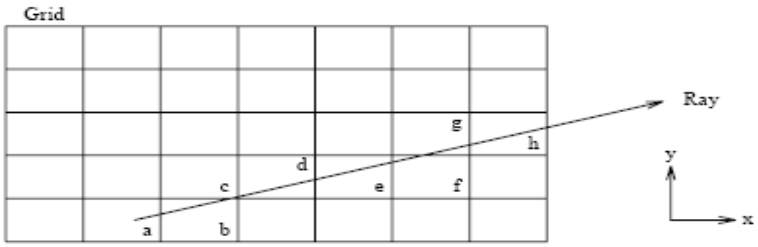
Voorbereiding vraagt eveneens typisch O (log n)… -> afweging

Faster and fewer intersections

- Bounding volume hierarchie
- Spatial partitioning in voxels
  - Default methode (*ref: Andrew Woo, John Amanatides*)
    - Opdeling van ruimte in niet overlappende gebieden (voxels)
      - Octrees, eventueel met variërende grootte (*ref: Glassner*)
      - 3D Grid partitioning (identieke grootte per voxel)
    - elke voxel bevat lijst met objecten die binnen deze voxel vallen
    - per voxel enkel intersecties onderzoeken van ray met objecten die behorende tot de lijst van die voxel

      - intersectie vastgesteld => berekening nieuwe reflected ray path
      - geen intersecties => traversal algoritme toepassen

    - Toepassing ray traversal algoritme: zoek naburige voxel
      vb: 3D DDA
      (3 dimensional digital difference analysis)



Faster and fewer intersections

Principe:

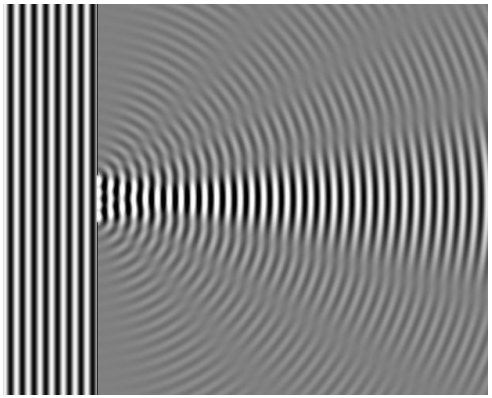
- **Toepassing bounding volume hiërarchie op vaste elementen buiten virtual volume**
  (mode stirrer, bijgevoegde absorbers, detailelementen Reverb Room, …)

- **Toepassen spatial partitioning in voxels voor virtual volume**
  per voxel eveneens data veldsterkte en polarisatie bewaren
  laat toe bepaling 3D E-veld

  Voordeel: reductie complexiteit tav eerste implementatie
  Nadeel: nog geen reductie omvang data in cache (cache overflow!)

Opm: **- structuur algoritmes: recursief, SIMD**

- Diffractie..?



Fewer rays

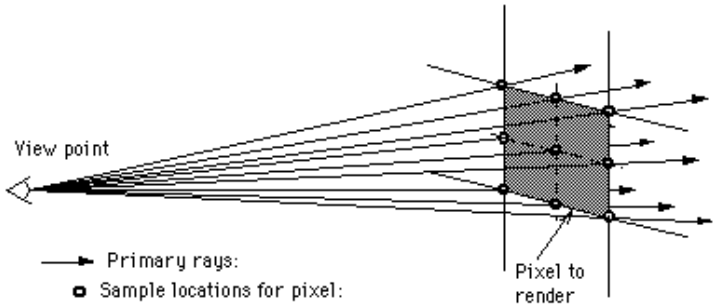
Werd reeds toegepast:

- **afbreken van ray bij intersectie van dempend materiaal ($\Gamma = 0$)**
- **afbreken van ray bij demping in functie van afgelegde weg (|E| ~1/r)**
- **Begrenzing op aantal intersecties**

Generalized rays

Beam, cone of pencil tracing

- **Bepalen of naburige rays, of rays binnen het frustrum een identiek pad afleggen**
- **voordeel: simulatie wave fronts**
- **Nadeel : beperkte verbetering complexiteit bij EM simulatie,**
  eerder gehanteerd bij illumination tgv diffuse reflecties



Optimal programming techniques

- Algoritmiek:
  - Dynamic programming ?
  - Didive & Conquer ?
  - …
- Gebruik optimale datastructuren ?
- …